MONGOEXPORT AND IMPORT

k. Anju Munoth

Export Data

- can use the mongoexport utility to export data from your MongoDB database, to a JSON or CSV file.

- The utility is located in the MongoDB bin directory (eg, /mongodb/bin). When you run the utility, provide the name of the database, the collection, and the file you want it to be exported to.

- To export data, first open a new Terminal/Command Prompt window, then type the applicable command.

Syntax

- mongoexport must be run directly from the system command line.

- **mongoexport --collection=<coll> [options]**

- Must specify the collection to export.
- If you do not specify an output file, mongoexport writes to the standard output (e.g. stdout).

Backup database with mongoexport

$ mongoexport

Export MongoDB data to CSV, TSV or JSON files.

options:

```
  -h [ --host ] arg         mongo host to connect to ( <set name>/s1,s2 for
  -u [ --username ] arg     username
  -p [ --password ] arg     password
  -d [ --db ] arg           database to use
  -c [ --collection ] arg   collection to use
  -q [ --query ] arg        query filter, as a JSON string
  -o [ --out ] arg          output file; if not specified, stdout is used
```

Host and port

- To specify a host and/or port of the MongoDB instance, can either:
- Specify the hostname and port in the --uri connection string:

```
        mongoexport --
uri="mongodb://mongodbo.example.com:27017/reporting"  --
collection=events  --out=events.json [additional options]
```

- If using the --uri connection string, specify the database as part of the string.
- Cannot use the command-line option --db in conjunction with the --uri connection string

Host and port

- Specify the hostname and port in the --host:
- **mongoexport --host="mongodbo.example.com:27017" --collection=events --db=reporting --out=events.json**

- Specify the hostname and port in the --host and --port:
- **mongoexport --host="mongodbo.example.com" --port=27017 --collection=events --db=reporting --out=events.json**

Connect to a Replica Set

- To connect to a replica set to export its data, you can either:
- Specify the replica set name and members in the --uri connection string:

-
  ```
  mongoexport --uri="mongodb://mongodb0.example.com:27017,mongodb1.example.com:27017,mongodb2.example.com:27017/reporting?replicaSet=myReplicaSetName" --collection=events --out=events.json
  ```
- Specify the replica set name and members in the --host:

-
  ```
  mongoexport --host="myReplicaSetName/mongodb0.example.com:27017,mongodb1.example.com:27017,mongodb2.example.com" --collection=events --db=reporting --out=events.json
  ```

Connect to a Replica Set

- By default, mongoexport reads from the primary of the replica set. To override the default, you can specify the read preference:
- Can specify the read preference in the --uri connection string

```
mongoexport --uri="mongodb://mongodb0.example.com:27017,mongodb1.example.com:27017,mongodb2.example.com:27017/reporting?replicaSet=myReplicaSetName&readPreference=secondary" --collection=events --out=events.json
```

- If specifying the read preference tags, include the readPreferenceTags option:

```
mongoexport --uri="mongodb://mongodb0.example.com:27017,mongodb1.example.com:27017,mongodb2.example.com:27017/reporting?replicaSet=myReplicaSetName&readPreference=secondary&readPreferenceTags=region:east" --collection=events --out=events.json
```

--fields=<field1[,field2]>, -f=<field1[,field2]>

- Specifies a field or fields to include in the export. Use a comma separated list of fields to specify multiple fields.
- If any of your field names include white space, use quotation marks to enclose the field list. For example, if you wished to export two fields, phone and user number, would specify --fields "phone,'user number'".
- For csv output formats, mongoexport includes only the specified field(s), and the specified field(s) can be a field within a sub-document.
- For JSON output formats, mongoexport includes only the specified field(s) and the _id field, and if the specified field(s) is a field within a sub-document, the mongoexport includes the sub-document with all its fields, not just the specified field within the document.

--fieldFile=<filename>

- An alternative to --fields.
- The --fieldFile option allows you to specify in a file the field or fields to include in the export and is only valid with the --type option with value csv.
- The file must have only one field per line, and the line(s) must end with the LF character (oxoA).
- mongoexport includes only the specified field(s). The specified field(s) can be a field within a sub-document.

--query=<JSON>, -q=<JSON>

- Provides a query as a JSON document (enclosed in quotes) to return matching documents in the export.
- Must enclose the query document in single quotes ('{ ... }') to ensure that it does not interact with your shell environment.
- Starting in MongoDB 4.2, the query must be in Extended JSON v2 format (either relaxed or canonical/strict mode), including enclosing the field names and operators in quotes:
- **mongoexport -d=test -c=records -q='{ "a": { "$gte": 3 }, "date": { "$lt": { "$date": "2019-01-01T00:00:00.000Z" } } }' --out=exportdir/myRecords.json**

--limit Option

- Limits the number of documents in the export.

mongoexport --db music --collection artists --limit 3 --out /data/dump/music/3_artists.json

- Resulting file:
- {"_id":{"$oid":"5780fbf948ef8c6b3ffb0149"},"artistname":"The Tea Party"}
- {"_id":{"$oid":"5781c9ac48ef8c6b3ffb014a"},"artistname":"Jorn Lande"}
- {"_id":1.0,"artistname":"AC/DC"}

--sort Option

- Specifies how the results are ordered.
- Here, we sort the file by the _id field in ascending order (i.e. 1). To make it descending, use a -1.

```
mongoexport --db music --collection artists --limit 3 --sort '{_id: 1}' --out /data/dump/music/3_artists_sorted.json
```

- Resulting file:
- {"_id":1.0,"artistname":"AC/DC"}
- {"_id":2.0,"artistname":"Prince","address":{"street":"Audubon Road","city":"Chanhassen","state":"Minnesota","country":"United States"}}
- {"_id":3.0,"artistname":"Moby","albums":[{"album":"Play","year":1999.0,"g

--skip Option

- Allows you to instruct mongoexport to skip a number of documents before starting the export operation.

mongoexport --db music --collection artists --limit 3 --sort '{_id: 1}' --skip 2 --out /data/dump/music/3_artists_sorted_skipped.json

- Resulting file:
- {"_id":3.0,"artistname":"Moby","albums":[{"album":"Play","year":1999.0,"genre":"Electronica"},{"album":"Long Ambients 1: Calm. Sleep.","year":2016.0,"genre":"Ambient"}]}
- {"_id":4.0,"artistname":"Rush"}
- {"_id":{"$oid":"5780fbf948ef8c6b3ffb0149"},"artistname":"The Tea Party"}

--pretty Option

Outputs documents in a more readable JSON format.

```
mongoexport --db music --collection artists --query
'{"artistname": "Miles Davis"}' --pretty --out
/data/dump/music/miles_davis_pretty.json
```

```
{
        "_id": {
                "$oid": "578214f048ef8c6b3ffb0159"
        },
        "artistname": "Miles Davis",
        "albums": [
                {
                        "album": "Kind of Blue",
                        "year": 1959.0,
                        "genre": "Jazz"
                },
                {
                        "album": "Bitches Brew",
                        "year": 1970.0,
                        "genre": "Jazz"
                }
        ]
}
```

Export all documents (all fields) into the file “domain-bk.json“.

```
$ mongoexport -d webmitta -c domain -o domain-bk.json
```

- connected to: 127.0.0.1
- exported 10951 records

Export all documents with fields “domain” and “worth” only

```
$ mongoexport -d webmitta -c domain -f "domain,worth" -o domain-bk.json
```

- connected to: 127.0.0.1
- exported 10951 records

Mongoexport with query

- Export all documents with a search query, in this case, only document with “worth > 100000” will be exported.

```
$mongoexport -d webmitta -c domain -f "domain,worth" -q '{worth:{$gt:100000}}' -o domain-bk.json
```

- connected to: 127.0.0.1
- exported 10903 records

Export a Collection to a JSON File

- Use mongoexport to export the artists collection to a JSON file:

```
mongoexport --db music --collection artists --out /data/dump/music/artists.json
```

- Resulting message:

```
2016-07-12T09:57:37.613+0700        connected to: localhost
2016-07-12T09:57:37.614+0700        exported 13 records
```

Export a Collection to a CSV File

- To export to a CSV file, add --type=csv to the command.
- Can also specify the fields in the MongoDB documents to export.
- Use mongoexport to export the artists collection to a CSV file. We export the _id and artistname fields. We've also given the file name a .csv extension.

```
mongoexport --db music --collection artists --type=csv --fields
_id,artistname --out /data/dump/music/artists.csv
```

- Resulting message:

```
2016-07-12T10:16:33.111+0700        connected to: localhost
2016-07-12T10:16:33.114+0700        exported 13 records
```

Export the results of a Query

- Can use the --query option to specify a query to export. The query must be enclosed in single quotes.
- Here, we export details on Miles Davis to a JSON file:

```
mongoexport --db music --collection artists --query '{"artistname": "Miles Davis"}' --out /data/dump/music/miles_davis.json
```

- Resulting message:

```
2016-07-12T10:32:19.794+0700        connected to: localhost
2016-07-12T10:32:19.795+0700        exported 1 record
```

MongoDB - Import Data

- MongoDB provides the mongoimport utility that can be used to import Extended JSON, CSV, or TSV files into a MongoDB database created by mongoexport, or potentially, another third-party export tool.
- mongoimport is located in the bin directory (eg, /mongodb/bin or wherever you installed it).

- To import data, open a new Terminal/Command Prompt window and enter mongoimport followed by parameters such as database name, collection name, source file name, etc.

Restore database with mongoimport

$ mongoimport

```
options:
  -h [ --host ] arg       mongo host to connect to ( <set name>/s1,s2 for sets)
  -u [ --username ] arg   username
  -p [ --password ] arg   password
  -d [ --db ] arg         database to use
  -c [ --collection ] arg collection to use (some commands)
  -f [ --fields ] arg     comma separated list of field names e.g. -f name,age
  --file arg              file to import from; if not specified stdin is used
  --drop                  drop collection first
  --upsert                insert or update objects that already exist
```

--host

- Default: localhost:27017
- Specifies a resolvable hostname for the mongod to which to connect.
- By default, the mongoimport attempts to connect to a MongoDB instance running on the localhost on port number 27017.
- To connect to a replica set, specify the replSetName and a seed list of set members, as in the following:
- --host <replSetName>/<hostname1><:port>,<hostname2><:port>,<...>
- When specifying the replica set list format, mongoimport always connects to the primary.

--username

- Specifies a username with which to authenticate to a MongoDB database that uses authentication.
- Use in conjunction with the --password and --authenticationDatabase options.

--password

- Specifies a password with which to authenticate to a MongoDB database that uses authentication.
- Use in conjunction with the --username and --authenticationDatabase options.
- To prompt the user for the password, pass the --username option without --password or specify an empty string as the --password value, as in --password "" .

--authenticationDatabase

- Specifies the authentication database where the specified --username has been created

--fields <field1[,field2]>,

- Specify a comma separated list of field names when importing csv or tsv files that do not have field names in the first (i.e. header) line of the file.
- To also specify the field type as well as the field name, use --fields with --columnsHaveTypes.
- If you attempt to include --fields when importing JSON data, mongoimport will return an error. --fields is only for csv or tsv imports.

--fieldFile <filename>

- As an alternative to --fields, the --fieldFile option allows you to specify a file that holds a list of field names if your csv or tsv file does not include field names in the first line of the file (i.e. header).
- Place one field per line.
- To also specify the field type as well as the field name, use --fieldFile with --columnsHaveTypes.
- If you attempt to include --fieldFile when importing JSON data, mongoimport will return an error. --fieldFile is only for csv or tsv imports.

--ignoreBlanks

- Ignores empty fields in csv and tsv exports. If not specified, mongoimport creates fields without values in imported documents.

- If you attempt to include --ignoreBlanks when importing JSON data, mongoimport will return an error. --ignoreBlanks is only for csv or tsv imports.

--type <json|csv|tsv>

- Specifies the file type to import. The default format is JSON, but it's possible to import csv and tsv files.
- The csv parser accepts that data that complies with RFC RFC 4180.
- As a result, backslashes are not a valid escape character.
- If you use double-quotes to enclose fields in the CSV data, you must escape internal double-quote marks by prepending another double-quote.

--file <filename>

- Specifies the location and name of a file containing the data to import.
- If you do not specify a file, mongoimport reads data from standard input (e.g. “stdin”).

--drop

- Modifies the import process so that the target instance drops the collection before importing the data from the input.

--headerline

- If using --type csv or --type tsv, uses the first line as field names.
- Otherwise, mongoimport will import the first line as a distinct document.
- If you attempt to include --headerline when importing JSON data, mongoimport will return an error. --headerline is only for csv or tsv imports.

--mode <insert|upsert|merge>

- Default: insert
- Specifies how the import process should handle existing documents in the database that match documents in the import file.
- By default, mongoimport uses the _id field to match documents in the collection with documents in the import file.
- To specify the fields against which to match existing documents for the upsert and merge modes, use --upsertFields.

--mode <insert|upsert|merge>

Value	Description
`insert`	Insert the documents in the import file. **mongoimport** will log an error if you attempt to import a document that contains a duplicate value for a field with a unique index, such as `_id`.
`upsert`	Replace existing documents in the database with matching documents from the import file. **mongoimport** will insert all other documents.
`merge`	Merge existing documents that match a document in the import file with the new document. **mongoimport** will insert all other documents.

--upsertFields <field1[,field2]>

- Specifies a list of fields for the query portion of the upsert.
- Use this option if the _id fields in the existing documents don't match the field in the document, but another field or field combination can uniquely identify documents as a basis for performing upsert operations.
- Modified in version 3.4: Modifies the import process to update existing objects in the database if they match based on the specified fields, while inserting all other objects. You do not need to use --mode upsert with --upsertFields.
- If you do not specify a field, --upsertFields will upsert on the basis of the _id field.
- To ensure adequate performance, indexes should exist for this field or fields.

--stopOnError

- Forces mongoimport to halt the insert operation at the first error rather than continuing the operation despite errors.

- Starting in version 4.2, mongoimport, by default, continues when it encounters duplicate key and document validation errors.
- To ensure that the program stops on these errors, specify --stopOnError.

--numInsertionWorkers <int>

- Default: 1
- Specifies the number of insertion workers to run concurrently.
- For large imports, increasing the number of insertion workers may increase the speed of the import.

--writeConcern <document>

- Default: majority
- Specifies the write concern for each write operation that mongoimport performs.
- Specify the write concern as a document with w options:
- --writeConcern "{w:'majority'}"

--bypassDocumentValidation

- Enables mongoimport to bypass document validation during the operation.
- This lets you insert documents that do not meet the validation requirements.

Examples

- Imports all documents from file “domain-bk.json” into database “myDb”collection named “emp”.
- All non-existing databases or collections will be created automatically.

```
$ mongoimport -d myDb -c emp --file domain-bk.json
```

- connected to: 127.0.0.1
- Wed Apr 10 17:26:12 imported 10903 objects

Mongoimport with --upsert

- With --mode upsert, mongoimport replaces existing documents in the database that match a document in the import file with the document from the import file.
- Documents that do not match an existing document in the database are inserted as usual.
- By default mongoimport matches documents based on the _id field. Use --upsertFields to specify the fields to match against

```
$ mongoimport -d myDb -c emp --file domain-bk.json --upsert
```

- connected to: 127.0.0.1

Import JSON File

- import collection back into database.

```
mongoimport --db music --file /data/dump/music/artists.json
```

- Resulting message:

```
2016-07-12T13:34:04.904+0700        no collection specified
2016-07-12T13:34:04.905+0700        using filename 'artists' as collection
2016-07-12T13:34:04.911+0700        connected to: localhost
2016-07-12T13:34:04.968+0700        imported 13 documents
```

Specify a Collection Name

- Can use the --collection argument to provide a name of the collection that the data should go into.

```
mongoimport --db music --collection jazz --file
/data/dump/music/miles_davis.json
```

- Resulting message:

```
2016-07-12T14:09:01.793+0700          connected to: localhost
2016-07-12T14:09:01.849+0700          imported 1 document
```

steps to import the CSV :

- **Step 1:** Go to the directory where MongoDB is kept.
- **Step 2:** Go to its bin folder.
- **Step 3:** Save or keep your CSV or TXT file here to which you want to import.
- This is our CSV file named **word.csv.**
- The top word is the **headerline** which will be used in the commandline MongoDb import command later.
- **Step 4:**
- Open a command prompt in the bin folder

steps to import the CSV :

- **Step 5:**
- Type in the under mentioned command.

```
mongoimport -d wordCupDictionary -c dailywords --type CSV --file
c:/users/user/desktop/word.csv --headerline
```

- And you will see that your document data will be imported.
- **Note : -> wordCupDictionary** is the database name
- **dailywords** is the collection name.
- **--type CSV** denotes that file type that is being imported is of type CSV.
- **word.csv** is the file name that we are importing.
- **--headerline** is the header of the CSV file, below which listed data is mentioned.(here in this example it is ‛ **word** ‛)

Import CSV File

- Can import a CSV file by using --type csv.
- If the CSV file has a header row, use --headerline to tell mongoimport to use the first line to determine the name of the fields in the resulting document.
- If the CSV file doesn't have a header row, use the --fields parameter to set the field names.

Import CSV File - With Header Row

- Here's an example of importing a document with a header row.

The contents of the CSV file:

```
_id,albumname,artistname
1,Killers,"Iron Maiden"
2,Powerslave,"Iron Maiden"
12,"Somewhere in Time","Iron Maiden"
3,"Surfing with the Alien","Joe Satriani"
10,"Flying in a Blue Dream","Joe Satriani"
11,"Black Swans and Wormhole Wizards","Joe Satriani"
6,"Out of the Loop","Mr Percival"
7,"Suck on This",Primus
8,"Pork Soda",Primus
9,"Sailing the Seas of Cheese",Primus
```

Import the file:

- **mongoimport --db music --collection catalog --type csv --headerline --file /data/dump/music/catalog.csv**
- Query the collection:
- > db.catalog.find()

```
{ "_id" : 2, "albumname" : "Powerslave", "artistname" : "Iron Maiden" }
{ "_id" : 1, "albumname" : "Killers", "artistname" : "Iron Maiden" }
{ "_id" : 3, "albumname" : "Surfing with the Alien", "artistname" : "Joe Satriani" }
{ "_id" : 12, "albumname" : "Somewhere in Time", "artistname" : "Iron Maiden" }
{ "_id" : 10, "albumname" : "Flying in a Blue Dream", "artistname" : "Joe Satriani" }
{ "_id" : 6, "albumname" : "Out of the Loop", "artistname" : "Mr Percival" }
{ "_id" : 7, "albumname" : "Suck on This", "artistname" : "Primus" }
{ "_id" : 8, "albumname" : "Pork Soda", "artistname" : "Primus" }
{ "_id" : 11, "albumname" : "Black Swans and Wormhole Wizards", "artistname" : "Joe Satriani" }
{ "_id" : 9, "albumname" : "Sailing the Seas of Cheese", "artistname" : "Primus" }
```

Import the file:Without Header Row

- Here's another CSV file, but this one doesn't have a header row:

```
Mutt Lange, 1948

John Petrucci, 1967

DJ Shadow, 1972

George Clinton, 1941
```

- Can import it and specify the field names to use:

```
mongoimport --db music --collection producers --type csv --
fields name,born --file /data/dump/music/producers.csv
```

Import JSON to Remote Host Running with Authentication

- mongoimport imports data from the file /opt/backups/mdb1-examplenet.json into the contacts collection within the database marketing on a remote MongoDB database with authentication enabled.
- mongoimport connects to the mongod instance running on the host mongodb1.example.net over port 37017.
- It authenticates with the username user; the example omits the --password option to have mongoimport prompt for the password:
- **mongoimport --host=mongodb1.example.net --port=37017 --username=user --collection=contacts --db=marketing --file=/opt/backups/mdb1-examplenet.json**

Points to remember

- Avoid using mongoimport and mongoexport for full instance production backups.
- Do not reliably preserve all rich BSON data types, because JSON can only represent a subset of the types supported by BSON.
- mongoimport only supports data files that are UTF-8 encoded. Using other encodings will produce errors.
- In order to connect to a mongod that enforces authorization with the --auth option, must use the --username and --password options.
- Connecting user must possess, at a minimum, the readWrite role on the database into which they are importing data.

MongoDB - Create a Backup

- To create a backup in MongoDB, either copy the files directly, or use one of several backup/management tools.

There are several ways to backup a MongoDB database:

- Copy the data files
- Use mongodump
- Use MongoDB Cloud Manager
- Use Ops Manager

Copy the Data Files

- Can copy the underlying data files that MongoDB uses to store data.
- These are located in the data directory.
- The default location of the data directory is /data/db, however, if you use a different location you will need to use that instead.
- Should copy the whole directory for a complete backup.
- Can also use snapshots if the volume supports it.
- For example, on Linux, use LVM (Logical Volume Manager) to create a snapshot, then you can copy from that snapshot to your backup site/remote location.

Basic mongodump Operations

- mongodump utility backs up data by connecting to a running mongod or mongos instance.
- Utility can create a backup for an entire server, database or collection, or can use a query to backup just part of a collection.
- When you run mongodump without any arguments, the command connects to the MongoDB instance on the local system (e.g. 127.0.0.1 or localhost) on port 27017 and creates a database backup named dump/ in the current directory.
- To backup data from a mongod or mongos instance running on the same machine and on the default port of 27017

Use mongodump

- Can use mongodump to backup the data and mongorestore to restore it.
- To quickly backup all contents of the running server, open a new Terminal/Command Prompt, change to a directory that you want the /dump folder to be created in, and type the following:
- **mongodump**

Resulting message:

```
2016-07-12T15:44:34.467+0700    writing music.artists to
2016-07-12T15:44:34.467+0700    writing music.musicians to
2016-07-12T15:44:34.467+0700    writing music.catalog to
2016-07-12T15:44:34.468+0700    done dumping music.artists (13 documents)
2016-07-12T15:44:34.469+0700    done dumping music.musicians (10 documents)
2016-07-12T15:44:34.469+0700    done dumping music.catalog (10 documents)
2016-07-12T15:44:34.470+0700    writing music.producers to
2016-07-12T15:44:34.470+0700    writing music.jazz to
2016-07-12T15:44:34.470+0700    done dumping music.producers (5 documents)
2016-07-12T15:44:34.470+0700    done dumping music.jazz (1 document)
2016-07-12T15:44:34.534+0700    writing test.restaurants to
2016-07-12T15:44:34.705+0700    done dumping test.restaurants (25359 documents
```

Backup a Single Database

- Can backup a single database by specifying the name of the database in the --db parameter:
- **mongodump --db=music**

Backup a Single Collection

- Can backup a single collection by specifying the name of the collection in the --collection parameter:

- **mongodump --db=music --collection=artists**

Specify a Backup Location

- Use the --out parameter to specify the directory that you'd like the backup to be written to:

- **mongodump --db music --out /data/backups**

Host and port

- Can also specify the --host and --port of the MongoDB instance that the mongodump should connect to.
- For example:

```
mongodump --host mongodb.example.net --port 27017
```

- mongodump will write BSON files that hold a copy of data accessible via the mongod listening on port 27017 of the mongodb.example.net host.

mongodump

- mongodump overwrites output files if they exist in the backup data folder.
- Before running the mongodump command multiple times, either ensure that you no longer need the files in the output folder (the default is the dump/ folder) or rename the folders or files.

Create Backups from Non-Local mongod Instances

- The --host and --port options for mongodump allow you to connect to and backup from a remote host.
- Consider the following example:

```
mongodump --host mongodb1.example.net --port 3017 --username user --password "pass" --out /opt/backup/mongodump-2013-10-24
```

- On any mongodump command you may, as above, specify username and password credentials to specify database authentication.

More Options

- mongodump has many more options for specifying how the data is backed up.
- Can always run mongodump --help to see which options are available.

Basic mongorestore Operations

- The mongorestore utility restores a binary backup created by mongodump. By default, mongorestore looks for a database backup in the dump/ directory.
- The mongorestore utility restores data by connecting to a running mongod or mongos directly.
- mongorestore can restore either an entire database backup or a subset of the backup.

Example

```
mongorestore dump-2017-10-25/
```

- Here, mongorestore imports the database backup in the dump-2017-10-25 directory to the mongod instance running on the localhost interface on the default port 27017.

Restore Point in Time Oplog Backup

- If you created your database dump using the --oplog option to ensure a point-in-time snapshot, call mongorestore with the --oplogReplay option, as in the following example:
- **mongorestore --oplogReplay**
- May also consider using the **mongorestore --objcheck** option to check the integrity of objects while inserting them into the database, or you may consider the **mongorestore --drop** option to drop each collection from the database before restoring from backups.

Restore Backups to Non-Local mongod Instances

- By default, mongorestore connects to a MongoDB instance running on the localhost interface (e.g. 127.0.0.1) and on the default port (27017).
- If you want to restore to a different host or port, use the --host and --port options.
- Consider the following example:

```
mongorestore --host mongodb1.example.net --port 3017 --username user
--password 'pass' /opt/backup/mongodump-2013-10-24
```

- As above, you may specify username and password connections if your mongod requires authentication.

MongoDB Cloud Manager

- MongoDB Cloud Manager is a hosted platform for managing MongoDB.
- Can use MongoDB Cloud Manager to continually back up MongoDB replica sets and sharded clusters by reading the oplog data from your MongoDB deployment.
- MongoDB Cloud Manager works on a subscription basis

Ops Manager

- Ops Manager is like MongoDB Cloud Manager, except it is installed on your local environment (i.e. not in the cloud).
- Can use it for monitoring, automating, and backup up your MongoDB deployment.
- Ops Manager is available to MongoDB Subscribers.

Mongodb Monitoring

- Monitoring is one of the most critical administrative activities in MongoDB. This is because you can be more proactive by monitoring the environment for possible issues which could crop up.

Below are some of the ways of implementing monitoring

- mongostat will tell you how many time database operations such as insert, query, update, delete, etc. actually occur on the server. This will give a good idea on how much the load the server is handling and will indicate whether you need additional resources on the server or maybe additional servers to distribute the load.
- mongotop tracks and reports the current read and write activity of a MongoDB instance, and reports these statistics on a per collection basis.
- MongoDB provides a web interface that exposes diagnostic and monitoring information in a simple web page. One can browse to the below url on your local server to open the web administration utility http://localhost:28017
- The serverStatus command, or db.serverStatus() from the shell, returns an overview of the status of the database, with details on the disk usage, memory use, connections established to the MongoDB environment, etc.

Administrative Command

JavaScript Database Administration Methods	Description
`db.cloneDatabase(<host>)`	Clone the current database from the `<host>` specified. The`<host>` database instance must be in noauth mode.
`db.copyDatabase(<from>, <to>, <host>)`	Copy the `<from>` database from the `<host>` to the `<to>`database on the current server. The `<host>` database instance must be in `noauth` mode.
`db.fromColl.renameCollection(<toColl>)`	Rename collection from `fromColl` to `<toColl>`.
`db.getCollectionNames()`	Get the list of all collections in the current database.
`db.dropDatabase()`	Drops the current database.

Opening Additional Connections

JavaScript Connection Create Methods	Description
db = connect("<host><:port>/<dbname>")	Open a new database connection.

conn = **new** Mongo()
db = conn.getDB("dbname")
	Open a connection to a new server using new `Mongo()`.
	Use `getDB()` method of the connection to select a database.

db.getSiblingDB()

- Return a reference to another database using this same connection without explicitly switching the current database.
- This allows for cross database queries.